SRA の技術情報誌
GSLetterNeo
Vol.187　2024 年 2 月

PyScript を使った Python による機械学習プログラムの作成

熊澤 努

技術本部 先端技術研究室

はじめに

GSLetterNeo Vol.176[1]では PyScript[2]を使い，HTML 上で Python を実行してみました．PyScript は Python で書かれたプログラムを Web ブラウザ上で実行するフレームワークです。JavaScript と同様に，Python プログラムを HTML に書いてブラウザで実行することができます．今回は Vol.176 の続きとして，Web ブラウザ Edge 上で Python の機械学習プログラムを動かしてみましょう．実行環境は Vol.176 と同じです．PyScript の使い方の詳細は公式サイトあるいは Vol.176 を参照してください．

[1] https://www.sra.co.jp/Portals/0/files/gsletter/pdf/GSLetterNeoVol176.pdf

[2] PyScript の最新の動向は公式サイトで確認してください．
https://pyscript.net/

機械学習プログラムについて

今回使う機械学習プログラムは XGBoost[3]です．XGBoost は決定木学習と呼ばれる方式の教師あり機械学習技術で，データがどのカテゴリに属しているか判定する分類問題に適しています．機械学習パッケージである scikit-learn[4]の分類問題用データセットに対して XGBoost を実行して，結果を HTML に表示するプログラムを書くことにしましょう．

XGBoost は Python の xgboost パッケージに実装されています．PyScript で使う場合は，scikit-learn，xgboost，データセットを格納するデータ構造のための pandas，後でグラフ表示のために使う matplotlib のインストールは不要です．これらは実行中にインストールされます．

今回書くプログラムは以下のサイトを元にしており，必要に応じて書き換えています．

https://qiita.com/predora005/items/19aebcf3aa05946c7cf4

https://di-acc2.com/programming/python/23577/

データセットを準備する

scikit-learn には分類問題のデータセットがいくつか用意されています．今回の記事では，iris データセット, breast cancer データセット，digits データセットを使います．ユーザはブラウザ上で分類したいデータセットを選択して，学習に必要な設定値を入力してから XGBoost を実行します．その結果，テストデータに対する XGBoost の正解率と，分類の際に重要視した特徴量を HTML で表示します．今回は学習に使う設定値として，XGBoost がつくる木構造の深さの最大値（max_depth:モデルの複雑さを表す指標）と，ブースティング回数（num_boost_round: 木構造を作る回数，学習の反復数に相当）をブラウザで設定します．

Vol.176 と同様に HTML ファイルを作成して，xgb.html という名前で保存しておきます．PyScript で必要なパッケージを扱うため，py-config は以下のように記述します．

```
<py-config>
    packages = ["matplotlib", "pandas", "scikit-learn", "xgboost"]
</py-config>
```

グラフ描画パッケージの matplotlib は特徴量の重要性をグラフで表示するために使用します．pandas パッケージはデータセットを格納するデータ構造として使用します．scikit-learn はデータセットとその前処理，正解率の測定に使います．

---

[3] XGBoost については以下の原論文を参照してください：Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In KDD '16, 785–794. https://doi.org/10.1145/2939672.2939785, https://arxiv.org/abs/1603.02754

[4] https://scikit-learn.org/stable/

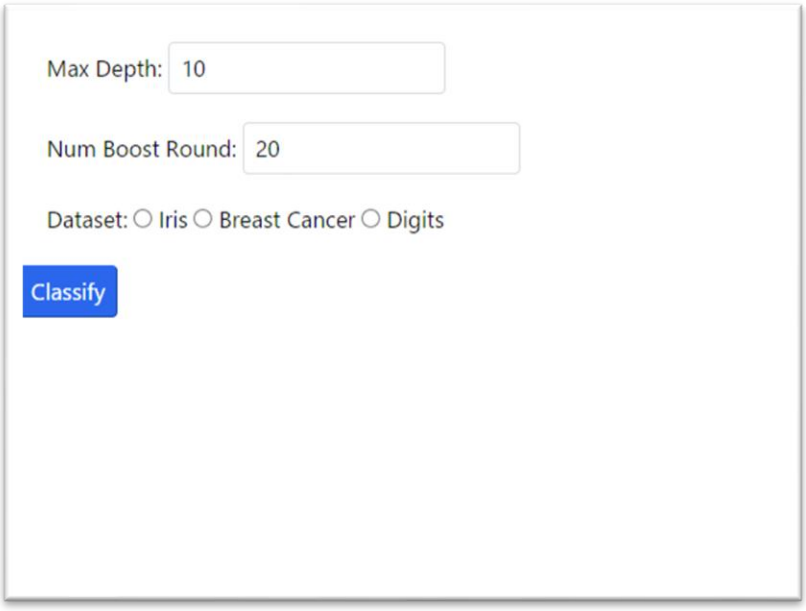
GUI 部品を記述する

xgb.html の body タグに GUI 部品を配置します．

```
<div id="md"  style="margin: 20px;">
    Max Depth:
    <input id="max_depth" class="py-input" value='10' type="text">
</div>
<div id="'md"  style="margin: 20px;">
    Num Boost Round:
    <input id="num_boost_round" class="py-input" value='20' type="text">
</div>
<div id="input" style="margin: 20px;">
    Dataset:
    <input type="radio" id="Iris" name="Dataset" value="Iris">
    <label for="Iris"> Iris</label>
    <input type="radio" id="Breast Cancer" name="Dataset" value="Breast Cancer">
    <label for="Breast Cancer"> Breast Cancer</label>
    <input type="radio" id="Digits" name="Dataset" value="Digits">
    <label for="Digits"> Digits</label>
</div>
<button id="classify" class="py-button" type="submit" py-click="classify()">
    Classify
</button>

<p id="score" style="font-size:20px;"></p>
<div id="graph-area"></div>
```

id が max_depth, num_boost_round であるテキストボックスが上で述べた XGBoost の設定値です．初期値をそれぞれ 10 と 20 にしておきます．その下にラジオボタンでデータセットを 1 つ選択するようにします． id を classify としたボタンで設定した値とデータセットを使って機械学習を実行します．score と graph-area は結果を表示する領域です．ここでつくった簡単なレイアウトを下の図に示します．



Python プログラムを記述する

それでは py-script タグに Python プログラムを記述します。

```
<py-script>
    import matplotlib.pyplot as plt
    import pandas as pd
    from sklearn.datasets import load_iris, load_breast_cancer, load_digits
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score
    import xgboost

    def make_dmatrix(dataset):
        x = pd.DataFrame(dataset.data, columns=dataset.feature_names)
        y = pd.Series(dataset.target)
        train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=0.2, shuffle=True)
        train = xgboost.DMatrix(train_x, label=train_y)
        test = xgboost.DMatrix(test_x, label=test_y)
        return train, test, test_y

    def classify():
        dataset_ids = js.document.getElementsByName("Dataset")
        selected = ''
        for id in dataset_ids:
            if id.checked:
                selected = id.value
                break

        if selected == 'Iris':
            dataset = load_iris()
            classes = 3
        elif selected == 'Breast Cancer':
            dataset = load_breast_cancer()
            classes = 2
        elif selected == 'Digits':
            dataset = load_digits()
            classes = 10
        else:
            raise NotImplementedError()

        max_depth = int(Element('max_depth').element.value)
        num_boost_round = int(Element('num_boost_round').element.value)
        param = {'max_depth':max_depth,'objective':'multi:softmax','num_class': classes}

        tr, test, test_y = make_dmatrix(dataset = dataset)

        model = xgboost.train(param, tr, num_boost_round)
        pred = model.predict(test)

        score = accuracy_score(test_y, pred)
        score_str = 'score:{0:.4f}'.format(score)
        pyscript.write('score',score_str)

        fig, ax = plt.subplots()
        xgboost.plot_importance(model, ax=ax)

        ax = xgboost.plot_importance(model)
        display(fig, target="graph-area", append=False)
</py-script>
```

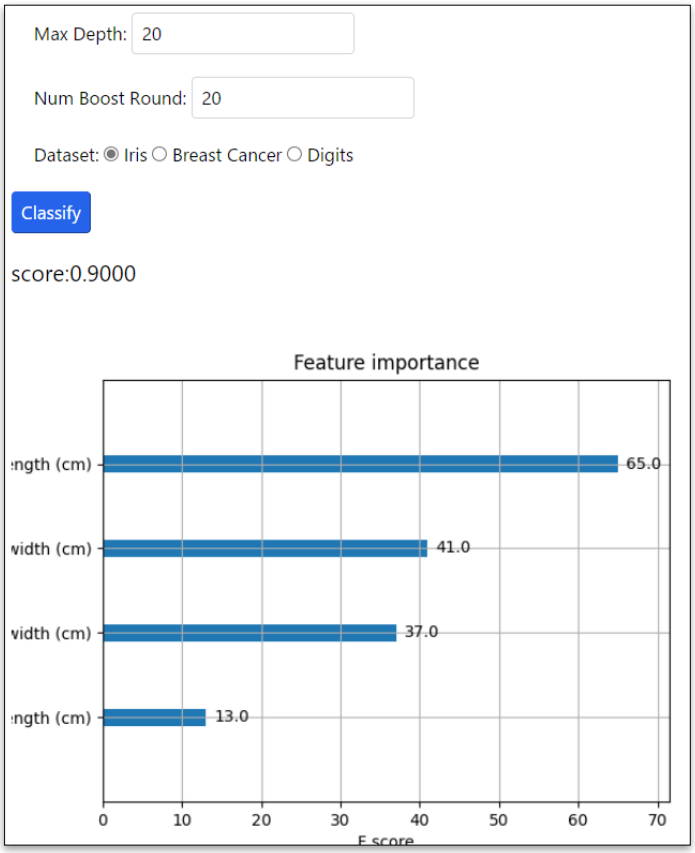
classify ボタンを押したときに発行されるイベントの処理が関数 classify です．

まず，各 GUI 部品から値を取得して，データセットや設定値を決定します．データセットは scikit-learn が提供する API を呼び出すだけで取得できます．if 文でラジオボタンの値に応じたデータセットのロードをするとともに，そのデータセットが持つカテゴリのクラス数を決定します．関数 make_dmatrix で，データセットを訓練用とテスト用に分割して，両者を XGBoost が扱うデータ構造 DMatrix インスタンスに変換します．なお，テストデータについては，訓練したモデルによる予測結果から正解率を算出するために，正解データを DMatrix インスタンスには変換せずに返すことにします．

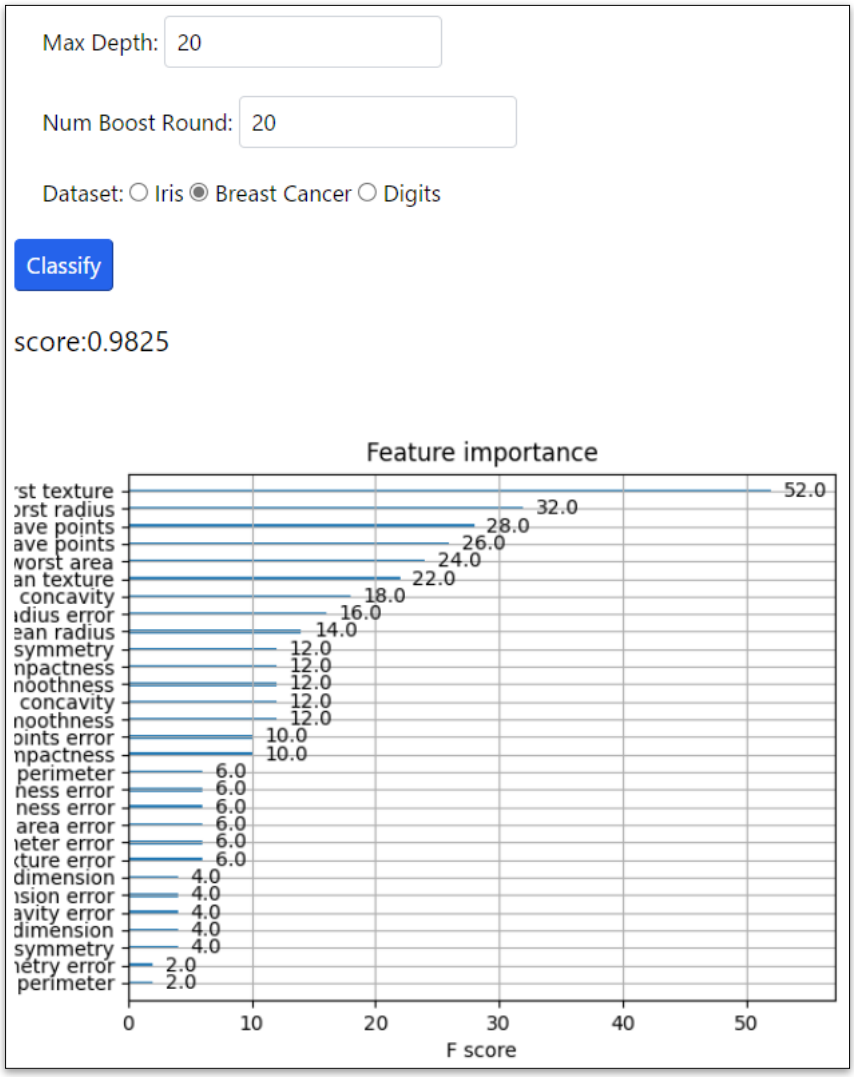
次に，XGBoost のモデルを生成します．xgboost の train 関数を呼び出すとモデルをつくるとともに，引数として与えたデータで訓練を行います．その結果，訓練済みのモデルが変数 model に代入されます．model の predict メソッドでテストデータに対する予測結果を算出します．正答率は scikit-learn の accuracy_score 関数で計算します．そして，xgboost の plot_importance 関数でデータの持つ各特徴量とその重要性（F 値）を算出してグラフにしています．最終行の display 関数で，生成したグラフを graph_area に出力します．

HTML を実行する

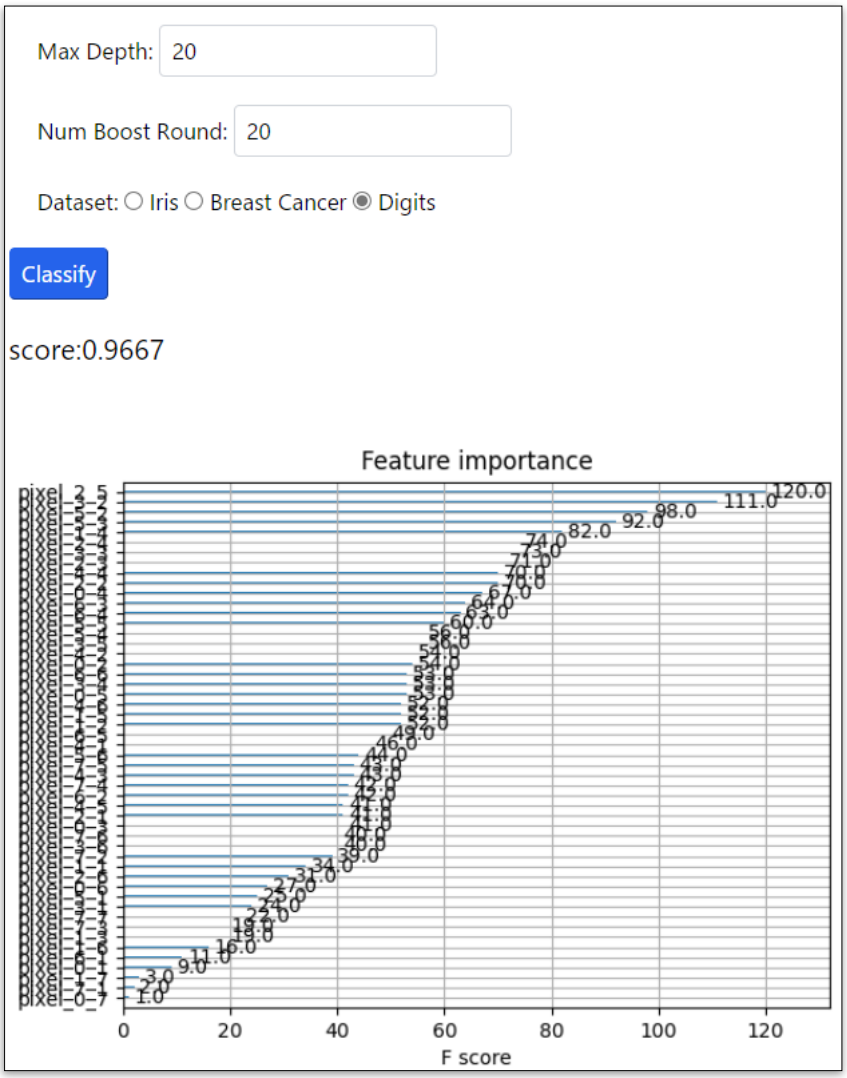
完成した HTML を Microsoft 社のブラウザである Edge で実行した結果を下の図に示します．この実行例は iris データセットに対する結果です．正解率は 90%で，前半が切れていますが，利用する特徴量にははっきりとした違いが見られることが分かります．



同じ設定値で，Breast Cancer データセットに対する結果も見てみます．正解率は約 98%で，高い汎化性能が得られていることが手軽に分かります．



Digits データセットに対する結果も示します．正解率は約 97%になりました．



xgb.html

最後に，少し長いためフォントが小さくなってしまいますが，完成した xgb.html を次のページに掲載します．

```
<html>
  <head>
    <title>XGBOOST Example</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="https://pyscript.net/latest/pyscript.css" />
    <script defer src="https://pyscript.net/latest/pyscript.js"></script>
  </head>
  <body>
    <py-config>
      packages = ["matplotlib", "pandas", "scikit-learn", "xgboost"]
    </py-config>
    <py-script>
        import matplotlib.pyplot as plt
        import pandas as pd
        from sklearn.datasets import load_iris, load_breast_cancer, load_digits
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        import xgboost

        def make_dmatrix(dataset):
            x = pd.DataFrame(dataset.data, columns=dataset.feature_names)
            y = pd.Series(dataset.target)
            train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.2, shuffle=True)
            train = xgboost.DMatrix(train_x, label=train_y)
            test = xgboost.DMatrix(test_x, label=test_y)
            return train, test, test_y

        def classify():
            dataset_ids = js.document.getElementsByName("Dataset")
            selected = ''
            for id in dataset_ids:
                if id.checked:
                    selected = id.value
                    break
            pyscript.write('score',id)

            if selected == 'Iris':
                dataset = load_iris()
                classes = 3
            elif selected == 'Breast Cancer':
                dataset = load_breast_cancer()
                classes = 2
            elif selected == 'Digits':
                dataset = load_digits()
                classes = 10
            else:
                raise NotImplementedError()

            max_depth = int(Element('max_depth').element.value)
            num_boost_round = int(Element('num_boost_round').element.value)
            param = {'max_depth': max_depth, 'objective': 'multi:softmax', 'num_class': classes}

            tr, test, test_y = make_dmatrix(dataset = dataset)

            model = xgboost.train(param, tr, num_boost_round)
            pred = model.predict(test)

            score = accuracy_score(test_y, pred)
            score_str = 'score:{0:.4f}'.format(score)
            pyscript.write('score',score_str)

            fig, ax = plt.subplots()
            xgboost.plot_importance(model, ax=ax)

            ax = xgboost.plot_importance(model)
            display(fig, target="graph-area", append=False)
    </py-script>

    <div id="'md"  style="margin: 20px;">
      Max Depth:
      <input id="max_depth" class="py-input" value='10' type="text">
    </div>
    <div id="md"  style="margin: 20px;">
      Num Boost Round:
      <input id="num_boost_round" class="py-input" value='20' type="text">
    </div>
    <div id="input" style="margin: 20px;">
      Dataset:
      <input type="radio" id="Iris" name="Dataset" value="Iris">
      <label for="Iris"> Iris</label>
      <input type="radio" id="Breast Cancer" name="Dataset" value="Breast Cancer">
      <label for="Breast Cancer"> Breast Cancer</label>
      <input type="radio" id="Digits" name="Dataset" value="Digits">
      <label for="Digits"> Digits</label>
    </div>
    <button id="classify" class="py-button" type="submit" py-click="classify()">
      Classify
      </button>

    <p id="score" style="font-size:20px;"></p>
    <div id="graph-area"></div>
  </body>
</html>
```

おわりに

今回は機械学習技術 XGBoost の Python での実装を，ブラウザ上で実行しました．HTML を使った実行環境には，jupyter notebook に代表されるノートブック方式がよく知られています．Python プログラムを HTML に書き込んでおく必要があるため，ノートブック形式のように手軽にはできないものの，Python プログラムを埋め込んだ HTML をプログラマ自身がつくることで，独自のレイアウトで機械学習の結果を表示できることが分かりました．

今回の記事を執筆した 2024 年 2 月現在では，PyScript と，Python を Web ブラウザで扱うために PyScript が使用する Pyodide パッケージ[5]は深層学習パッケージである tensorflow や pytorch をサポートしていません．そのため，複雑な機械学習技術を実行することには適していないかもしれません．しかし，statsmodels のような統計パッケージはサポートしているため，統計的推測技術をブラウザ上で動かすには十分な技術と考えてよいでしょう．

[5] https://pyodide.org/en/stable/

GSLetterNeo Vol.187
2024 年 2 月 20 日発行
発行者　株式会社 SRA 技術本部 先端技術研究室

編集者　熊澤努　方 学芬
バックナンバー https://www.sra.co.jp/public/sra/gsletter/
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8